Electronic Communications of the EASST
Volume 67 (2014)

Proceedings of the 13th International Workshop on Graph Transformation and Visual Modeling Techniques (GTVMT 2014)

Modelling Adaptive Networks: The Case of the Petrified Voters

Mudhafar Hussein, Reiko Heckel, Vincent Danos, Pawel Sobocinski

12 pages

Guest Editors: Frank Hermann, Stefan Sauer
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Modelling Adaptive Networks: The Case of the Petrified Voters

Mudhafar Hussein[1], **Reiko Heckel**[1], **Vincent Danos**[2], **Pawel Sobocinski**[3]

[1] mshh2@le.ac.uk | reiko@mcs.le.ac.uk
Department of Computer Sciences, Leicester University, UK
[2] vincent.danos@gmail.com
School of Informatics, University of Edinburgh, UK
[3] sobocinski@gmail.com
School of Electronics and Computer Science, University of Southampton, UK

Abstract: Adaptive networks are characterised by mutual dependencies between nodes' local state changes and evolving topology. Stochastic graph transformation systems are ideally suited to model such networks, but in order to analyse their properties we require more scalable methods. We present a case study of a simple but representative example of adaptive networks. In this social network of opinionated voters a node connected to another of different opinion will either convert (changing state) or disconnect and establish a new connection with a node of the same opinion (changing topology).

To analyse quantitative properties of the model, such as the long-term average ratio of edges connecting nodes of different opinions or the overall rate of change of opinions or connections, we use a refinement technique developed for the Kappa graph rewriting approach to derive a stochastic Petri net, replacing graphs as states by markings representing the frequency of occurrences of certain patterns. In general the number of patterns (and therefore places) is unbounded, but approximations can be used to replace complex patterns by combinations of simpler ones.

Keywords: stochastic graph transformation, refinement, stochastic Petri net, adaptive networks, social network analysis

1 Introduction

Graph transformation systems are a natural model for networks with evolving topology, representing network nodes and edges by graph vertices, attributed to model local states, and graph edges, respectively. Adaptive Networks are characterised by mutual dependencies between topology evolution and local state changes in network nodes [GB08]. For example, a state change may depend on a node's connections while in turn enabling the creation of new connections that could lead to changes in other nodes, etc.

This interdependency prevents a layered view, where state and topology changes occur at different speeds and are handled at different levels, such as in traditional approaches to dynamic reconfiguration of component-based systems where the components' communication and local computation is halted in order for reconfiguration to take place. Adaptive networks do not allow this two-level approach and are therefore intrinsically more complex. They are also stochastic

systems, where operations are equipped with probability distributions governing delays. Examples range from biological systems and social networks, peer-to-peer or other distributed systems with autonomous components, to technical networks such as power grids, the Internet and mobile communication infrastructure [GB08]. Modelling such networks by attributed graphs, both the evolution of the network topology and local state changes can be specified by rules while stochastic graph transformations allow us to represent probabilistic timing.

However, as with most complex models a major challenge is the scalability of their analysis. In this paper we describe an approach to abstraction that approximates a stochastic graph transformation model by a stochastic (place-transition) Petri net. Its places represent graph patterns, with tokens as pattern occurrences, while rules are modelled by net transitions. Nets can be analysed more efficiently and scalably by simulation or computing the steady state solutions of the underlying Markov chains.

In order to represent a rule as a net transition, the rule has to have a deterministic effect on the patterns represented by the net's places, creating and destroying a fixed number of occurrences of all patterns. This is not the case in general, where the effect of the rule on certain patterns may depend on the context of the application. A system whose rules are deterministic in this sense is called *balanced* and, assuming limits on the degree of the graphs which are also preserved by the rules, a graph transformation system can be transformed into an equivalent balanced one. However, this technique does not guarantee that a rule's applicability can be captured exactly unless its left-hand side is a pattern itself. We consider an approximation of larger patterns by combinations of more basic ones (represented by places of the net), statistically valid only on large random graphs.

Balanced refinements are inspired by Kappa where they play a crucial role in enabling analytical techniques such as the thermodynamic approach [DHH13] and the derivation of differential equations [FDH+09, DHJS13]. The idea is to replace each rule by a set of extensions that are jointly equivalent (in the sense of a stochastic bisimulation) to the original rule, such that the extended rules describe all the cases in which the original rule could affect any of the chosen patterns. This is possible due to a property of graphs, called *rigidity*, similar to the absence of V-structures in graphs [Dör95] where no node is allowed to carry two or more edges unless they are distinguishable by their types or attributes, or those of their target nodes. To control embeddings of rules into context and preserve rigidity of graphs under rule application, we employ reflection constraints based on the concept of open maps [Hec12]. Finally, to keep the patterns to be observed basic, and so limit the number of places of the Petri net, we develop statistical approximations of the more complex graph patterns arising as left-hand sides.

The approach will be illustrated by a small example of a dynamic social network exploring the interdependency between voting behaviour and relationships of individuals. The model will be introduced in Section 2. After some background in Section 3 it will be refined and petrified in Section 4, before Section 5 discusses related work and Section 6 ends the paper.

2 A Social Voter Model

We work with typed attributed graph transformations following the DPO approach [EEPT06]. A type graph provides us with a vocabulary for instance graphs. However, depending on what

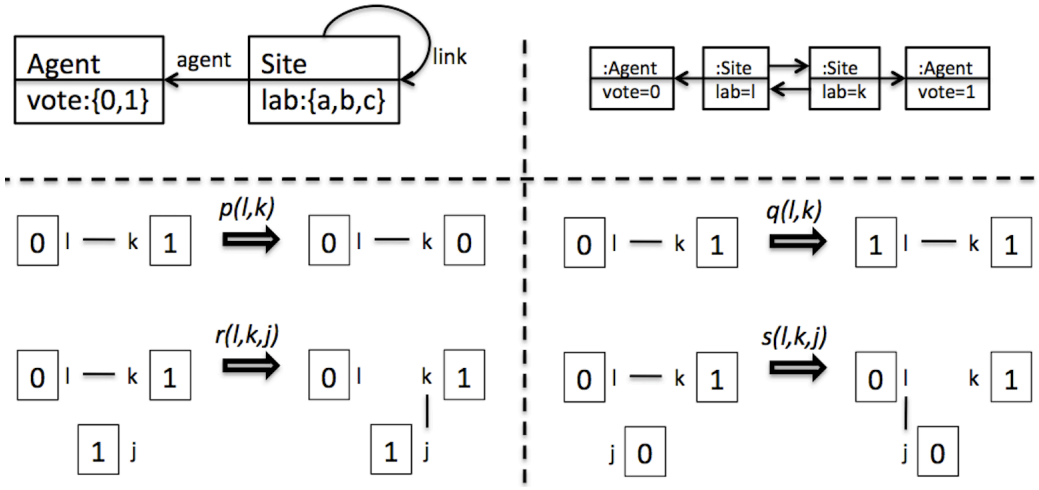


Figure 1: type graph (top left), instance graph (top right) and rules in concrete syntax (bottom) where $l,k,j \in \{a,b,c\}$

graphs are meant to represent, i.e., states or patterns, they are subject to further constraints. *States* are the most constrained, negatively by stating the absence of certain structures or positively requiring their presence. *Patterns* forming, e.g., the left- and right-hand sides of rules, are subject to negative constraints only, because they represent fragments of states, not deemed to be complete.

The model, adapted from [DGL+12], describes the behaviour of opinionated agents which vote for one of two parties and are less likely to stay friendly in case they disagree. The type graph is shown in the top left of Fig. 1. We represent votes as node attributes $0,1$. Connections are identified by labels a,b,c on the sites they are attached to. Once restricted to rigid graphs, this will limit the number of an agent's connections to 3. In the lower part of Fig. 1 rules are given in a condensed Kappa-like notation.

Durett's original model [DGL+12] is based on simple, symmetric graphs whose degree is bounded by 3 for analysis purposes, and where rules are described informally in English. The present model replaces this system by a Kappa-like rewriting system on *site graphs*, i.e., graphs over the metamodel in the top left of Fig. 1, where sites control and identify the attachment of links to Agent nodes. Such an encoding is always possible for a system of bounded degree.

Vote attributes are shown as labels inside Agents. Sites connected to Agents are shown by their labels only. Each Site label is attached to exactly one Agent, leaving agent edges implicit. Link edges are assumed to be symmetric, shown as undirected. To compare, the top right of Fig. 1 shows the left-hand side of rule p as full instance graph, omitting edge types, which can be inferred from sources and targets. Rules are given by rule schemata, e.g., $p(l,k)$ represents all rules obtained by choosing for l,k any labels from a,b,c. That means, l,k are not variables in the sense of attributed graphs, instantiated by matches, but metavariables to express rule schemata.

Rules model the coevolution of votes and connections: If two connected agents hold different votes, either one is converted to the opinion of the other (rules p,q), or the link between them is

broken and one makes a new connection to an agent of the same opinion (rules r, s).

For each of the rules shown we also assume its reverse rule. These are needed in order to ensure that the labelled transition system is strongly connected, and so leads to an irreducible Markov chain. In order to reflect their relative urgency, rules are equipped with rates as shown in the listing below. Rules labelled by "*" represent inverse rules, i.e., p^* is the inverse of p.

```
p = 10    q* = 1       q = 10    p* = 1
r = 20    r* = 1       s = 20    s* = 1
```

For example, p is 10 times as fast as p^* (in other words: if either rule is applicable in a graph, the average delay of p is one tenth of that of p^*) while r is twice as fast again. In particular, in a situation where two agents of different opinion are connected, splitting up is more likely than converting one's peer or oneself. One might think, just looking at the rates, that splitting up is twice as likely, but this is only true if there is exactly one free candidate in the graph for reconnection. If there are two alternative candidates, the number of matches doubles, increasing the probability for a match for r or s to be chosen as opposed to a match for p or q.

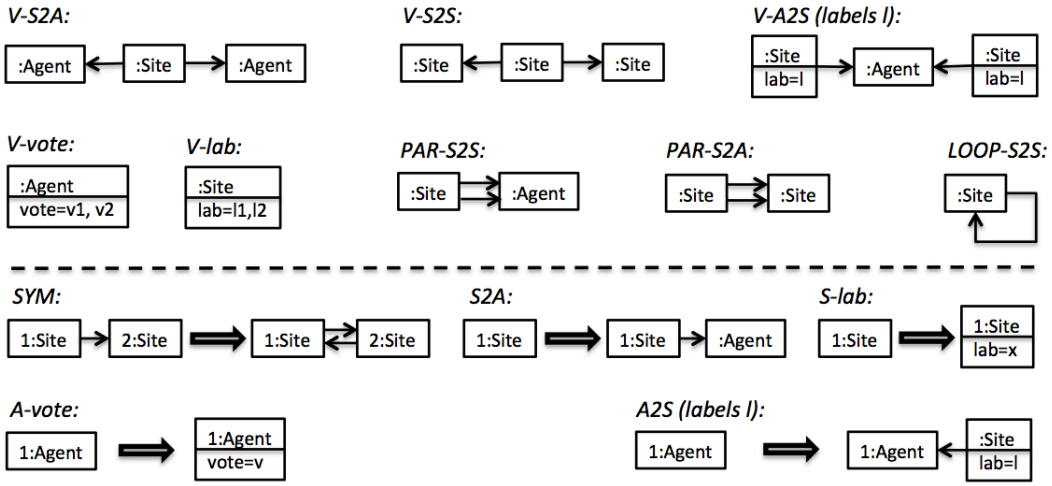


Figure 2: constraints on patterns (top) and states (bottom)

Pattern and state constraints are given in the top and bottom of Fig. 2, resp. *Patterns* are subject to negative constraints, expressed by forbidden patterns. They require the absence of V-structures, parallel edges and loops, i.e., *V-S2A, V-S2S:* no Site is connected to two Agents nor Sites, *V-A2S:* no Agent is connected to two Sites with the same label, *V-vote:* no Agent has two vote attributes, *V-lab:* no Site has two labels and *PAR-S2A, PAR-S2S, LOOP-S2S:* there are no parallel edges or loops.[1] States are graphs satisfying constraints *SYM:* link edges are symmetric, *S2A:* every Site is connected to an Agent, *S-lab:* each Site has a label attribute, *A-vote:* each Agent has a vote, *A2S:* for all labels $l \in \{a, b, c\}$, each Agent has a Site labelled l. Again, l is a metavariable expanding *A2S* into three concrete constraints.

[1] Unlike UML object diagrams whose notation we adopt, attributed graphs can have multiple values for the same node attribute. This is why we need *V-vote* and *V-lab* to rule out such cases.

For the model presented we are interested in questions such as: What is the evolution over time of the number of agents holding certain votes, or of edges connecting agents of the same vs. those of different votes? What are the resulting long-term ratios? The refinement of the model and its translation into a Petri net in the following section provide the prerequisite of such analyses.

3 Rigid Graph Transformation with Reflection Constraints

Refinement is aided by restrictions on the classes graphs and matches. *Rigid graphs* are without V-structures, such as specified by the negative pattern constraints in the top of Fig. 2. Reflection constraints act as rule-independent negative application conditions. This section gives an informal account of both restrictions, motivating their relevance for refinements.

Rule refinements are built from refinements of patterns first, extending them canonically to spans over patterns representing rules. A pattern A can be seen as a predicate over states validated by a match $a : A \to G$. Refining the pattern, we want to replace A by a set of extensions $e_i : A \to E_i$, each representing a stronger predicate, such that their exclusive disjunction is equivalent to A: The set of solutions for A should split into disjoint subsets of solutions for the extended patterns. In other words, for every occurrence $a : A \to G$ there should be a *single extension* $e_i : A \to E_i$ with a *unique decomposition* of a as $b_i \circ e_i$ for $b_i : E_i \to G$. This ensure that matches (and therefore transformations) for the original and the refined rules into a given graph are in one-to-one correspondence.

The identity on A satisfies these requirements. In order for refinements to be nontrivial, we restrict graphs and matches. The key to there being exactly one extension covering each case is the restriction of matches by reflection constraints, formally matches that are *open maps* [Hec12].

The reflection constraint in the left of Fig. 3 states that there should not be links attached to an agent's sites in a graph if there are no such links in a pattern (e.g., the rule's left-hand side). As before, this is a family of constraints covering all possible labels. Consider the instance *R-AS2S(a,b)* in the centre. The (crossed-out) match of the *new* rule on the right does not reflect the link attached to the *a*-labelled site in the given graph, i.e., the link is in the graph but not in the left-hand side of the rule. A morphism of patterns that satisfies the reflection constraints is called *reflective*. A transformation using this illegal match would lead to the crossed-out graph in the bottom right, which violates the pattern constraint *V-S2S* of Fig. 2. In general, reflection constraints define minimal forbidden examples for matches, in analogy to forbidden subgraphs for graph morphisms.

The uniqueness of $b_i : E_i \to G$ in the decomposition of $a : A \to G$ as $b_i \circ e_i$ is due to *rigidity*, which ensures that partial matches extend uniquely, or not at all, to larger (connected) patterns. The forbidden pattern *V-A2S* in Fig. 2 provides an example of a non-rigid graph, where extensions of partial matches are not unique. Assume A to be the graph with a single Agent and let B consist of an Agent and connected Site labelled l. The only match match $m : A \to$*V-A2S* has two different extensions $f, g : B \to$*V-A2S* using the left or right Site in *V-A2S* to map the Site in B.

A *rigid graph transformation system* (RGTS) is given by a set of of rules R over an attributed type graph TG together with sets of pattern, state, and reflection constraints C, S, X, respectively, such that C ensures rigidity and transformations along reflective matches preserve state and

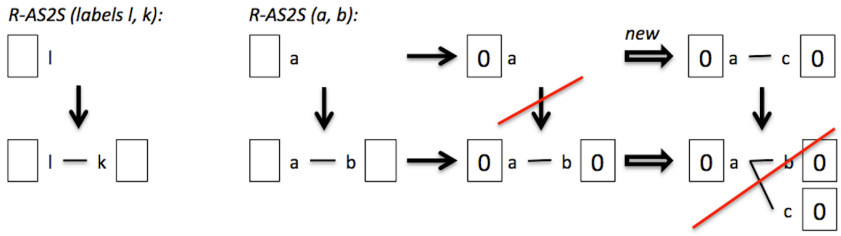


Figure 3: Reflection constraint ensuring preservation of pattern constraints *V-S2S*

pattern constraints. All our rules in Fig. 1 preserve the constraints but, as can be shown in analogy to the example in Fig. 3, reflectivity of the match is a necessary condition.

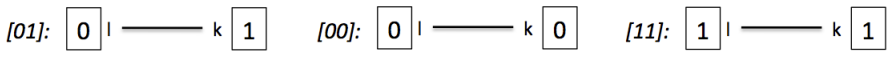
In a stochastic RGTS each rule r in R is associated with a rate constant written $k(r)$. R is *finite-state* for a given initial graph G over TG if the state space accessible from the initial state is finite up to isomorphism. This data determines a finite continuous-time Markov chain (CTMC) on graphs. It will be convenient to write X_t for the associated random variable which describes the (random) state of the system at time t.

4 Petrify and Approximate

Our journey from stochastic RGTS to stochastic Petri net proceeds in three steps: First, we define a set of pattern observables P, which will form the places of the net. Then we refine the rules R by R' such that (1) the new rules are equivalent (stochastically bisimilar) to the old set and (2) each rule in R' is P-balanced, i.e., creates and destroys a fixed number of occurrences for each pattern in P. The rules in R' can then be regarded as the transitions of the net. Finally, we turn to the definition of rate expressions, i.e., transition rates that depend on the number of token on their input places. This will be based on an approximation of the number of occurrences of complex patterns, such as rules' left-hand sides, by simpler patterns in P.

An *observable* is a real-valued function on state graphs. Among observables, some are of specific interest to us, namely the *pattern observables*. Such an observable is specified entirely by a graph Q. Given a graph X, $[Q](X)$ is counting the number of injective reflective morphisms from Q into X. This notion extends readily to linear combinations of pattern observables, which we will also call pattern observables. Mostly, we are interested in the case where Q is connected.

We select as our basic observables the patterns $[00]$, $[01]$, $[11]$ counting, respectively, the 00, 01, 11 edges in a graph.



However, as before, these patterns are really pattern schemes, unfolded by instantiating site label variables l,k by actual labels a,b,c. Thus, the observables of interests are linear combinations of the form

$$[00] = \sum_{l,k\in\{a,b,c\}} [0(l^1),0(k^1)]$$

denoting by $[0(l^1),0(k^1)]$ a pattern of two agents voting 0 connected by a link 1 via sites labelled l,k. This choice of observables is natural for the Durett model, especially the 01 conflictual edges. Having said that, the method will work regardless of whether the chosen patterns have semantic significance.

A *pattern refinement* is an injective reflective morphism of patterns that intersects all components of its (rigid) target graph.[2] A *refinement of a rule* r with left-hand side L in a rigid GTS is a set of pattern refinements $e_i : L \to L_i$ satisfying r's gluing conditions, such that for every injective reflective $a : L \to G$ there is a single embedding $e_i : L \to E_i$ with a unique decomposition $a = b_i \circ e_i$ for $b_i : L_i \to G$. As a result

- each such pattern refinement e_i gives rise to a refined rule r_i by applying r to L_i along e_i
- for every transformation $G \overset{r,m}{\Longrightarrow} H$ via r there exists exactly one refined rule r_i and match m_i such that $m = m_i \circ e_i$ and $G \overset{r_i,m_i}{\Longrightarrow} H$.

The reverse is true, i.e., an application of the refined rule gives rise to a unique application of the original rule r. A refinement of a rule therefore does not only preserve the transformation relation but establishes a one-to-one correspondence between the transformations using r and its set of refined rules r_i. That means, given a rate for r in a stochastic rigid graph transformation system, replacing r by its refinement will not change the stochastic behaviour if all rules in the refinement inherit r's rate.

We are interested in constructing refinements that are *P-balanced* with respect to a given set of patterns P, i.e., where, for all patterns $p \in P$, applications of a refined rule r_i to a state graph $G \overset{r_i,m_i}{\Longrightarrow} H$ create and destroy a fixed number of occurrences of p. That means, we are able to assign to r_i a function $\Delta(r_i) : P \to \mathbb{Z}$ giving us for each pattern $p \in P$ an integer $\Delta(r_i)(p)$ representing the number of occurrences added (if positive) or destroyed (if negative). A system of balanced rules for a given set of patterns P yields a Petri net N with places P and transitions R' consuming and producing tokens as defined by the function $\Delta : R \times P \to \mathbb{Z}$, which becomes the incidence matrix of the net.

Let us apply this idea to our case study. A P-balanced refinement for rule $p(l,k)$ of Fig. 1 is shown in Fig. 4. The matches for $p(l,k)$ factor uniquely through exactly one of the extensions. Since matches reflect links, if an agent in a rule shows a site that is not connected, this site cannot be connected in the state graph. For example, the rule in the top right cannot be applied to a graph where the agent voting 1 has any further connections, because all three sites are present. Instead, the agent voting 0 in the same rule can have two more connections on the sites not mentioned in the rule. Each of the rules describes exactly one embedding of the original p into an immediate context.

Depending on the context, a rule may destroy or create different numbers of occurrences of patterns. The refinement in Fig. 4 replaces the rule $p(l,k)$ with a set of jointly equivalent rules, each balanced with respect to $[01],[00],[11]$. This balance is given below the rule arrow, e.g., the rule in the top right destroys one occurrence of $[01]$ and creates two of $[00]$. (Occurrences of $[11]$ or $[00]$ always come in pairs because of their symmetry.)

[2] A component $C \subseteq B$ is a disjoint subgraph, i.e., there exists a D with $C+D$ isomorphic to B. Intersection of $h(A) \subseteq B$ with components $C \subseteq B$ can be characterised categorically by the pullback of C's coproduct injection with h.

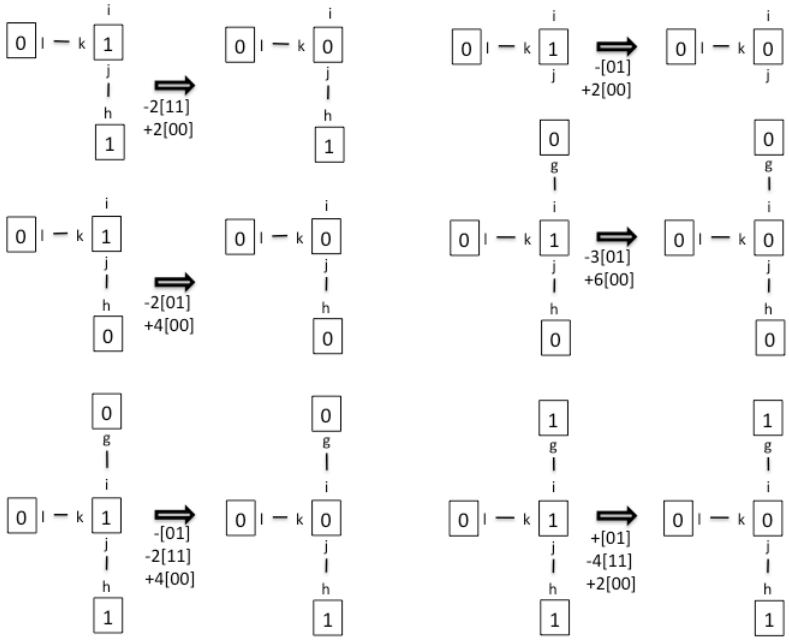


Figure 4: Refinement for rule $p(l,k)$. Labels $l,k,j,i,h,g \in \{a,b,c\}$; i,j,k pairwise distinct. Below each rule, the change in the number of occurrences of the 3 patterns is indicated.

The refinement in Fig. 4 is derived systematically by considering how pattern occurrences are affected by application of rules. That means, for every overlap of a subrule with a pattern such that elements of the pattern are created or destroyed, the rule is extended to include the pattern entirely. As a construction, this is a variation of critical pair analysis between $p(l,k)$ and the identity rule on any of the patterns. It follows the algorithm in [DHH13] for the Kappa language, where the correctness for this procedure is established.

In a Petri net N where places represent patterns, transitions can thus encode the effect of refined rules R' on the number of pattern occurrences. We also have a projection map π from the states of R' to the markings of N. However, the net's true dynamics should coincide with the projection of that of the initial rule set: $Y_t = \pi(X_t)$. The challenge is to express this dynamics Y_t entirely in terms of the states of N. To do this *exactly* is impossible, as the rate $\lambda_r(X_t)$ of a given rule in R', does not only depend on the projection. In other words, there is in general no λ'_r such that $\lambda_r(X_t) = \lambda'_r(\pi(X_t))$. Yet in other words, the projection π does not generate a forward stochastic bisimulation of CTMCs [DP03].

To see this, consider an instance of the middle left rule in Fig. 4:

$$r' = 0(a^1), 1(a^1, b^2, c), 0(a^2) \to 0(a^1), 0(a^1, b^2, c), 0(a^2)$$

again using superscripts to indicated which sites are linked with one another.

The rate of r depends on the constant $k(r)$ *and* on the number of ways in which the lhs of r can be embedded in the current state. That is to say it depends on $[0(a^1), 1(a^1, b^2, c), 0(a^2)]$ And this value cannot always be reconstructed uniquely from the projection.

Below, we will write, e.g., $[01_20]$ for the linear combination obtained by unfolding the above observable, with the subscript indicating the the degree of the middle node. We can approximate $[01_20]$ in terms of our extant observables as follows:

$$[01_20] \sim \frac{[01][1_20]}{[1_{\geq 1}]} \sim \frac{[01]}{[1_{\geq 1}]} \cdot \frac{[1_2][10]}{[1_{\geq 1}]} = \frac{[01]^2[1_2]}{[1_{\geq 1}]^2}$$

To derive the above, we use a breakdown principle: given a connected Q as a union of two graphs Q_1, Q_2, we approximate $[Q]$ as:

$$[Q] \sim \frac{[Q_1][Q_2]}{[Q_1 \cap Q_2]}$$

Probabilistically, this amounts to saying that $[Q]_1(X_t)$ and $[Q]_2(X_t)$ are *conditionally independent* given $[Q_1 \cap Q_2](X_t)$. Note that in the concrete breakdown, we have introduced new smaller observables which we now see are the intersection terms, e.g., $[1_{\geq 1}]$ which is short for the obvious linear combinations $\sum_{d \geq 1}[1_d]$.

The decomposition above is akin to the concept of *pair approximation* which is common in statistical physics and the study of complex networks [Gle13]. The quality of this type of approximation will depend on the particular model at hand, and will generally, but not always, become asymptotically exact as the underlying graph size goes to infinity, which is when one needs it most.

Indeed, when evaluating the expressions over the small graphs in Fig 5, the graph on the left demonstrates that this is not always accurate, because $[01] = 3, [1_2] = 2$ and $[1_{\geq 1}] = 3$ yielding an approximation of $3^2 \cdot 2/3^2 = 2$ while there is no single occurrence of $[01_20]$. The graph on the right yields $3^2 \cdot 0/5^2 = 0$ and has indeed no match for $[01_20]$.

Given these approximations, the rate expression for the transition representing rule r is $k(r) \cdot [01]^2[1_2]/[1_{\geq 1}]^2$, based on the rate constant $k(r) = k(p(l,k))$ of the refined rule r as inherited from the original. Applying this procedure to all rules in Fig. 1 as well as their reverse rules, we obtain a Petri net of 11 places and 60 transitions approximating the graph transformation system, with 6 transitions each arising as refinements of $p(l,k)$ and $q(l,k)$ and their inverse rules, and 9 each for $r(l,k,j)$ and $s(l,k,j)$ and their reverse rules.

5 Related Work

The aim of this paper is to investigate the abstraction of graph transformation systems by place-transition Petri nets, replacing the complex state representation of graphs by the much simpler one of a vector of natural numbers. This is motivated by the scalability of analyse methods to large systems. Work on abstraction in graph transformation has followed a variety of approaches and motivations. In [PP95], it is a means to improve comprehensibility of complex GTS by hiding and retrieving substructures as required. To enable analysis of models, many approaches aim at reducing the state space or behaviour representation. [BBKR08] uses *neighbourhood abstraction* to group graph elements via an equivalence relation up to some radius defining a node's neighbourhood. This allows the level of precision to be adjusted if the current abstraction does not allow the verification of properties. [YTTH08] uses a similar approach,

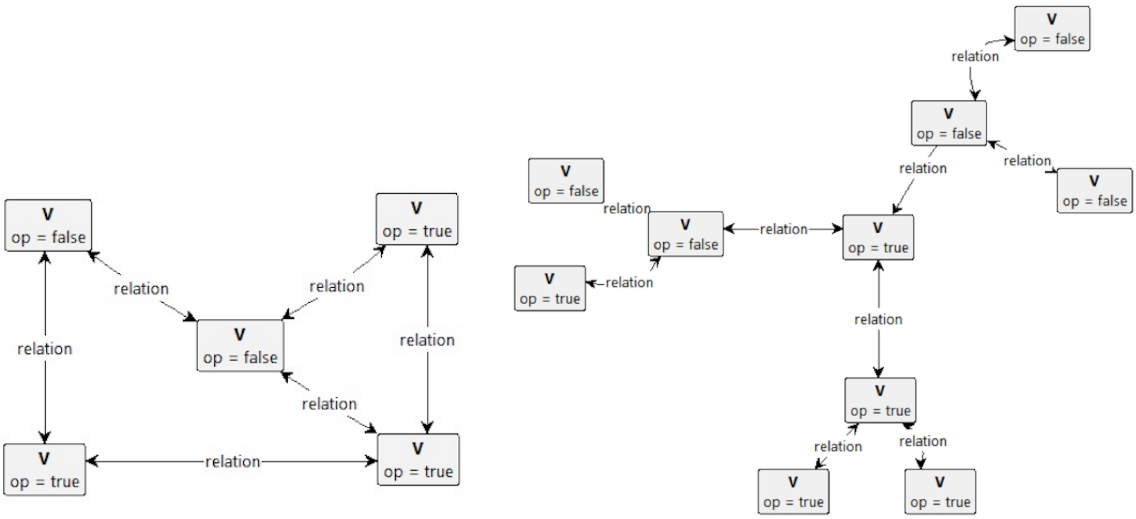


Figure 5: Two sample graphs.

but abstracted nodes are characterised by satisfaction of temporal logic formulae representing some behavioural property of the concrete system. In [RD06], based on shape graphs introduced in [SRW98], nodes are grouped by structural similarity with multiplicities to capture concrete representations of an abstract *shape*. Several states are therefore combined into a single structure. In counterexample-guided abstraction refinement based on unfoldings [KK06], the behaviour of a GTS is represented by a Petri graph representing an approximated unfolding.

All but the last approach keep within the same formalism, replacing one graph transformation system by a more abstract one. The last approach [KK06] derives a Petri net-like representation of the unfolding, i.e., at the semantic level, while we translate the specification into a Petri net. Indeed, as mentioned in the Introduction, in terms of methodology our approach has more in common with the analysis techniques of Kappa [FDH+09, DHJS13]. In particular, our transitions with rate expressions capturing pattern occurrences are reminiscent of the mass action semantics of stochastic Petri nets in [DO13].

6 Conclusion

We have presented a case study in analysing adaptive networks based on a methodology to reduce a given graph transformation system (over rigid graphs and with open maps as matches) to an equivalent Petri net. The central concept is that of a balanced refinement, which is constructed systematically by overlapping patterns and rules. Such a refinement, if it exists, guarantees the correctness of the Petri net and thus allows us to verify properties of the original system pertaining to numbers of occurrences of certain patterns P by analysing the corresponding Petri net. The numbers of tokens on places correspond to numbers of occurrences of patterns.

The evaluation suggests that the approach is correct in the sense that the stochastic behaviour is preserved and that the net-based analysis is more scalable than that of the original GTS. How-

ever, constructing a balanced refinement is itself a demanding task, which requires automation to scale to models of more interesting size. Another potential bottleneck is the state space analysis on the constructed Petri net.

For categories other than graphs we do not know under which conditions balanced refinements can effectively be constructed. A general theorem demonstrating the correctness of the approach is under investigation and can be the basis for automating the refinement and mapping. The generalisation from graphs to suitable adhesive categories would allow for attributed graphs over numerical data types, significantly extending the ability to model complex adaptive networks. Exploring further applications in this rich domain is another goal for the future.

Another new area of research may be the transformation of random graphs and the study of the statistical approximation of their complex patterns by combinations of simpler ones. Apart from the present application, similar approximations have been used to derive differential equations for adaptive networks, providing a scalable method to analyse the dynamics of large systems.

Bibliography

[BBKR08] J. Bauer, I. Boneva, M. Kurbán, A. Rensink. A modal-logic based graph abstraction. *Graph Transformations*, pp. 321–335, 2008.

[DGL+12] R. Durrett, J. Gleeson, A. Lloyd, P. Mucha, F. Shi, D. Sivakoff, J. Socoloar, C. Varghese. Graph fission in an evolving voter model. *Proceedings of the National Academy of Science* 109:3682–3687, 2012.

[DHH13] V. Danos, R. Harmer, R. Honorato-Zimmer. Thermodynamic Graph Rewriting. In *24th International Conference on Concurrency Theory, CONCUR'13*. LNCS 8052, pp. 380–394. Springer-Verlag, Aug. 2013.

[DHJS13] V. Danos, R. Honorato-Zimmer, S. Jaramillo-Riveri, S. Stucki. Deriving rate equations for site graph rewriting systems. In *Workshop on Static Analysis and Systems Biology, SASB, Seattle*. 2013.

[DO13] V. Danos, N. Oury. Equilibrium and termination II: the case of Petri nets. *Mathematical Structures in Computer Science* 23(2):290–307, 2013.

[Dör95] H. Dörr. *Efficient Graph Rewriting and Its Implementation (Lecture Notes in Computer Science)*. Springer-Verlag, 1995. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387600558

[DP03] J. Desharnais, P. Panangaden. Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. *J. Log. Algebr. Program.* 56(1-2):99–115, 2003.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer,

2006.
http://www.springer.com/3-540-31187-4

[FDH+09] J. Feret, V. Danos, R. Harmer, J. Krivine, W. Fontana. Internal coarse-graining of molecular systems. *PNAS* 106(16):6453–8, Apr. 2009.

[GB08] T. Gross, B. Blasius. Adaptive coevolutionary networks: a review. *J. R. Soc. Interface* 5(20):259–271, 2008.
doi:10.1098/rsif.2007.1229

[Gle13] J. P. Gleeson. Binary-State Dynamics on Complex Networks: Pair Approximation and Beyond. *Phys. Rev. X* 3:021004, Apr 2013.
doi:10.1103/PhysRevX.3.021004
http://link.aps.org/doi/10.1103/PhysRevX.3.021004

[Hec12] R. Heckel. DPO Transformation with Open Maps. In Ehrig et al. (eds.), *ICGT*. LNCS 7562, pp. 203–217. Springer, 2012.

[KK06] B. König, V. Kozioura. Counterexample-guided abstraction refinement for the analysis of graph transformation systems. *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 197–211, 2006.

[PP95] F. Parisi-Presicce, G. Piersanti. Multilevel graph grammars. In *Graph-Theoretic Concepts in Computer Science*. Pp. 51–64. 1995.

[RD06] A. Rensink, D. Distefano. Abstract graph transformation. *Electronic Notes in Theoretical Computer Science* 157(1):39–59, 2006.

[SRW98] M. Sagiv, T. Reps, R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 20(1):1–50, 1998.

[YTTH08] M. Yamamoto, Y. Tanabe, K. Takahashi, M. Hagiya. Abstraction of graph transformation systems by temporal logic and its verification. *Verified Software: Theories, Tools, Experiments*, pp. 518–527, 2008.